Multiplanarity —
a model for dependency structures in treebanks

Anssi YLI-JYRÄ

The PhD programme of Nordic Funding for Advanced studies (NorFA)

Department of General Linguistics, University of Helsinki

The KIT graduate school for language tecnology, Finland

TLT 2003, Växjö

1 – Overview

1. DEPENDENCY SYNTAX:
   - basic concepts
   - does not capture linear word order (without augmentations)
2. Simple hypotheses for WORD-ORDER:
   - projectivity and planarity
3. Finding a word-order in SYNTACTIC “WONDERLANDS”:
   - the idea of multiplanarity
   - a restriction motivated by the formal language theory
   - modeling treebanks: the algorithm and the coverage evaluation
4. Concluding remarks and possible APPLICATIONS

2 – Dependency Syntax

- An approach to syntax based on links between words which occur together (adapted from Hudson (1984))
- Abstracts away from the linear word-order in the surface strings
- Goes back to Tesnière (1959), and even to the Middle-Ages

3 – Dependency Syntax

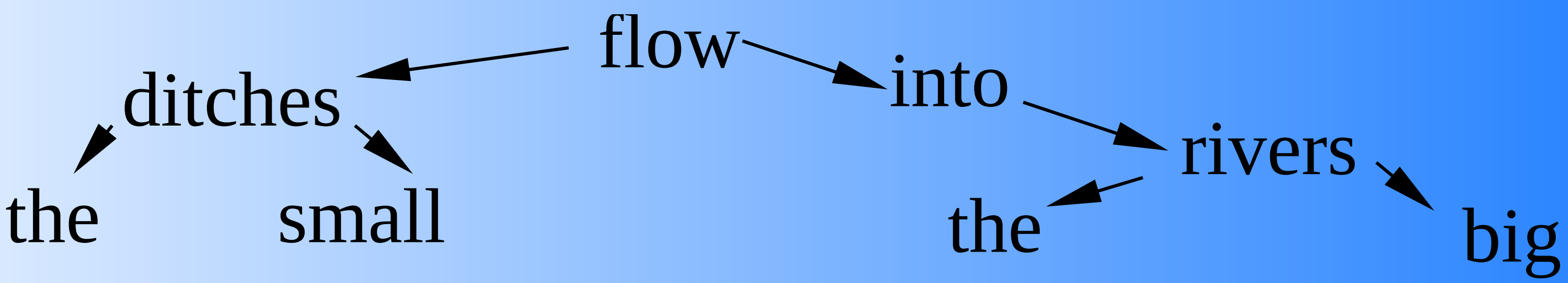
- Basic concepts:
  - in the context, each word (or nucleus) has a *syntactic function*;
  - words can be modified or complemented by other words (nuclei)
    $\Rightarrow$ binary *dependency relation* (“**governs**”) over words;

3 – Dependency Syntax

- Basic concepts:
  - in the context, each word (or nucleus) has a *syntactic function*;
  - words can be modified or complemented by other words (nuclei)
    $\Rightarrow$ binary *dependency relation* (“**governs**”) over words;
- Assumption of dependency **trees**:
  - a unique *root* word;
  - other words are governed by exactly one other word;
  - all the words are governed by the root $\rightarrow$ *acyclic* and *connected* graph

4 – Dependency syntax and linear word-order

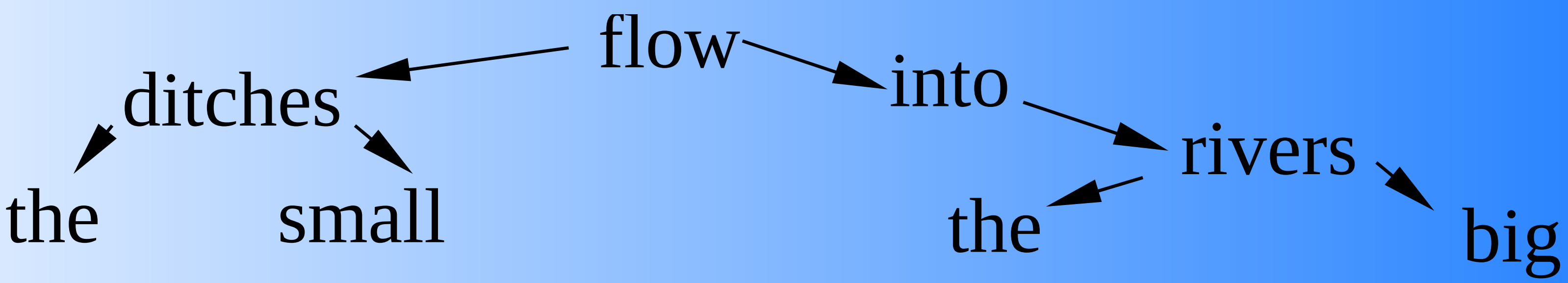
- An example of a dependency syntactic tree:



- The dependency trees do not necessarily show the actual word-order (compare to deep structures in transformational grammar)
- Description of the relation between dependency trees and the linearly ordered word sequences requires an extension to dependency syntax:

4 – Dependency syntax and linear word-order

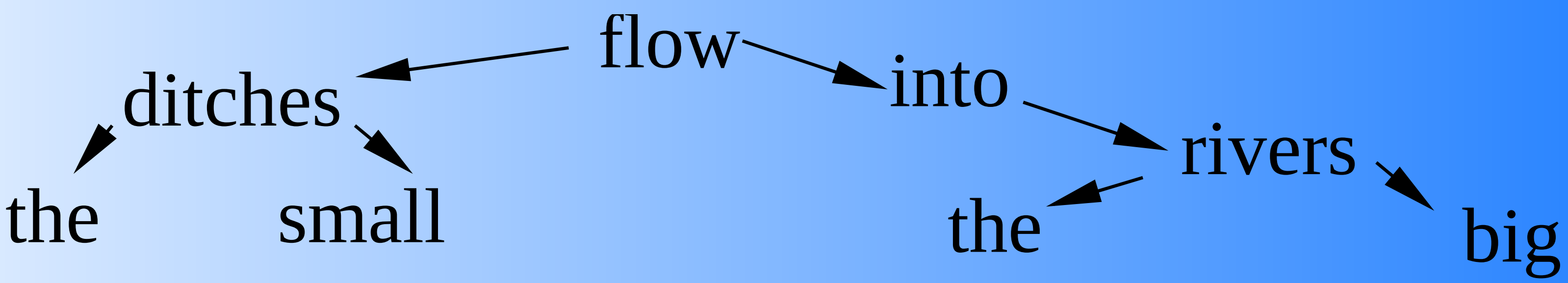
- An example of a dependency syntactic tree:



- The dependency trees do not necessarily show the actual word-order (compare to deep structures in transformational grammar)
- Description of the relation between dependency trees and the linearly ordered word sequences requires an extension to dependency syntax:
  - **simple hypotheses**: various kinds of *projectivity*

4 – Dependency syntax and linear word-order

- An example of a dependency syntactic tree:



- The dependency trees do not necessarily show the actual word-order (compare to deep structures in transformational grammar)

- Description of the relation between dependency trees and the linearly ordered word sequences requires an extension to dependency syntax:
  - **simple hypotheses**: various kinds of *projectivity*
  - **lexicalized constraints**: word-order domains, topological fields, etc.

5 – The simple hypotheses: definition

Let $< W_1, W_2, \ldots, W_n >$ be a sequence of words in the surface word-order.

The undirected (symmetric) version of the dependency relation **governs** is denoted by **link**.

5 – The simple hypotheses: definition

Let $< W_1, W_2, \ldots, W_n >$ be a sequence of words in the surface word-order.

The undirected (symmetric) version of the dependency relation **governs** is denoted by **link**.

Projective if, for any linked words W_i and W_j, $i < j$, it holds that if there is an intervening word W_k, $i < k < j$, then either W_i or W_j transitively **governs** W_k.

5 – The simple hypotheses: definition

Let $< W_1, W_2, \ldots, W_n >$ be a sequence of words in the surface word-order.

The undirected (symmetric) version of the dependency relation **governs** is denoted by **link**.

Projective if, for any linked words W_i and W_j, $i < j$, it holds that if there is an intervening word W_k, $i < k < j$, then either W_i or W_j transitively **governs** W_k.

Planar if, for any linked words W_i and W_j, $i < j$, it holds that if there is an intervening word W_k, $i < k < j$ then either W_i or W_j is transitively **linked to** W_k.

6 – The simple hypotheses: usefulness

In many languages (French, German, Italian, Danish, English, Russian, Swedish, Finnish, Czech), a large percentage of the sentences are projective (Lecerf 1960, etc.).

According to Marcus (1967), there are about 117 000 structured strings which may be formed with seven given words, but only 3876 of these are projective.

In other words, projectivity-like hypotheses has a lot of practical importance.

7 – Towards generalized projectivity: planarity

A **plane** is a space with two dimensions. **Planar graph** is a graph that can drawn in a plane without crossing edges.

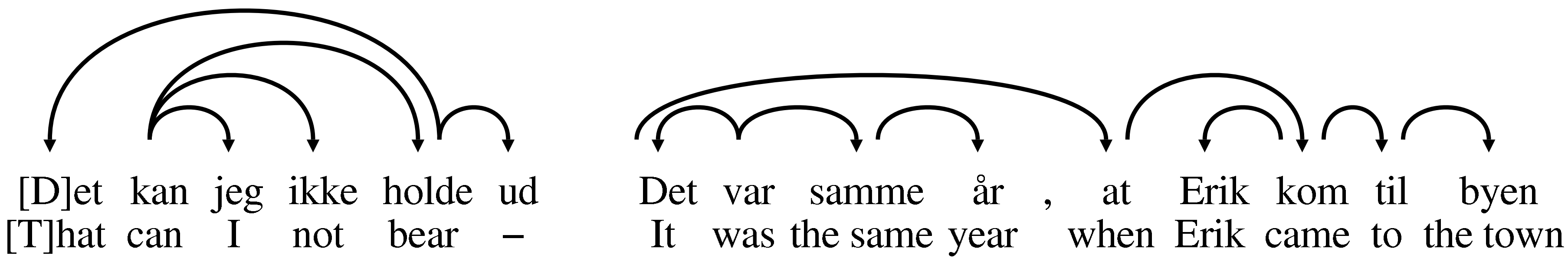
Planar dependency graph of a sentence is a dependency graph that can be drawn above the sentence without crossing dependency links.

7 – Towards generalized projectivity: planarity

A **plane** is a space with two dimensions. **Planar graph** is a graph that can drawn in a plane without crossing edges.

Planar dependency graph of a sentence is a dependency graph that can be drawn above the sentence without crossing dependency links.

The following trees are are planar but they are not projective (Figure 1 in the paper).



Moreover, these trees can be made projective by changing the direction of one dependency link only (“holde”—“kan”, “det”—“var”).

8 – Capturing the word-order in “syntactic wonderlands”

The Research Problem: Quite often extrapositions result into word-orders where the obtained dependency analyses are nonprojective and nonplanar.

Is there a generalized projectivity property that captures important word-order regularities in non-planar sentences?

8 – Capturing the word-order in “syntactic wonderlands”

The Research Problem: Quite often extrapositions result into word-orders where the obtained dependency analyses are nonprojective and nonplanar.

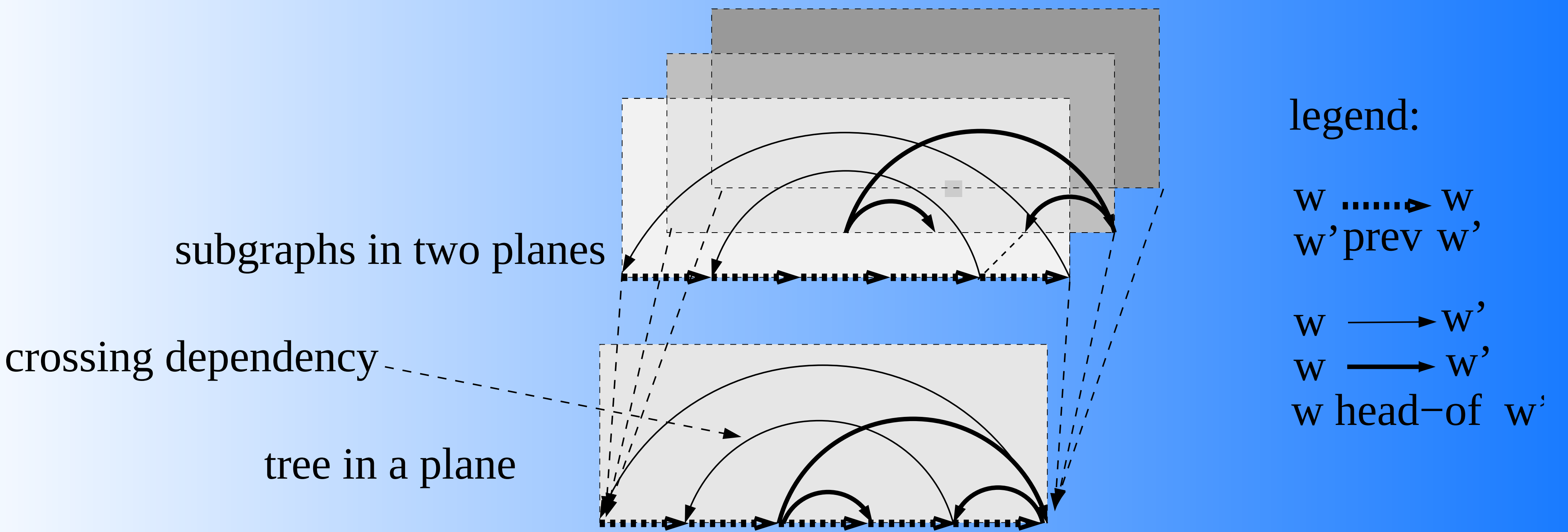
Is there a generalized projectivity property that captures important word-order regularities in non-planar sentences?

Approach:

- start from an overshoot solution: unrestricted multiplanarity
- make a restriction motivated by the formal language theory
- evaluating the linguistic adequacy of the restricted multiplanarity

9 – Multiplanarity

Multiplanar dependency graph is a finite union of planar dependency graphs drawn above the same sentence.



10 – A restriction motivated by the formal language theory

10 – A restriction motivated by the formal language theory

- It is generally believed that the language theoretic complexity of natural languages does not exceed the complexity of the so called *mildly context-sensitive languages* (Joshi 1985).

10 – A restriction motivated by the formal language theory

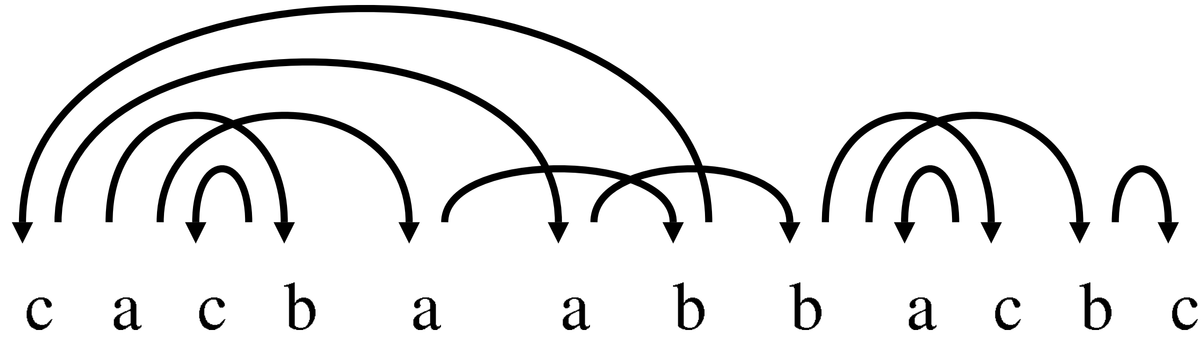
- It is generally believed that the language theoretic complexity of natural languages does not exceed the complexity of the so called *mildly context-sensitive languages* (Joshi 1985).
- The set of 2-planar trees capture a set of ordered dependency trees that yields the so-called MIX language.

10 – A restriction motivated by the formal language theory

- It is generally believed that the language theoretic complexity of natural languages does not exceed the complexity of the so called *mildly context-sensitive languages* (Joshi 1985).
- The set of 2-planar trees capture a set of ordered dependency trees that yields the so-called MIX language.
- The MIX language is not contained into mildly context-sensitive languages; it is more complex

$\Rightarrow$ We need to restrict multiplanar trees somehow in order to make the multiplanar structures linguistically more interesting.

11 – An example: a two-planar MIX sentence



·	·	[?^{1}		a^1][?^{1}		b^1]	·	·	·	·	·
[c^1						?^{1}]	·	[?^{1}		b^1]	·
·	·	[c	?]	·	·	·	·	·	·	·	·
·	[?		b]	·	·	·	·	[a	?]	·	·
[?					a][?		b][?		c]	[?	c]

Legend: The superscript ^{1} is used for links in the upper plane; [? is used in the head side of each link.

Observe: Both the planes start and end dependency links quite freely.

12 – Restricted multiplanarity: alignment constraints

Assume (i) that the words are processed from left to right (i.e. the planes **start** and **end** links at certain time moments), and (ii) when needed, additional planes $0, 1, 2, \ldots$ are introduced, and the plane with a lower number is used is used if possible.

A restricted version of multiplanarity is expressed by means of the **Plane Locking** constraint (and some other constraints):

> Links can always end regardless of the plane which they belong to. However, plane p cannot start a new link if there is a plane r, $r > p$ such that plane r contains an unended link.

13 – Distributing links to multiple planes: an algorithm

Input: $\langle W_1, W_2, ..., W_n \rangle$, and a set of dependency links

Output: a restricted multiplanar representation for the dependency structure

13 – Distributing links to multiple planes: an algorithm

Input: $\langle W_1, W_2, ..., W_n \rangle$, and a set of dependency links
Output: a restricted multiplanar representation for the dependency structure

Initially, assert that all the links belong to the plane number 0.

13 – Distributing links to multiple planes: an algorithm

Input: $\langle W_1, W_2, ..., W_n \rangle$, and a set of dependency links

Output: a restricted multiplanar representation for the dependency structure

Initially, assert that all the links belong to the plane number 0.

for foreach i in $[1..n-1]$ and j in $[i+1..n]$

  if there is a link connecting words W_i and W_j

    denote the plane where the link between W_i and W_j belongs to with p

13 – Distributing links to multiple planes: an algorithm

Input: $\langle W_1, W_2, ..., W_n \rangle$, and a set of dependency links
Output: a restricted multiplanar representation for the dependency structure

Initially, assert that all the links belong to the plane number 0.
for foreach i in $[1..n-1]$ and j in $[i+1..n]$
    if there is a link connecting words W_i and W_j
        denote the plane where the link between W_i and W_j belongs to with p
        if, there are two linked words W_k and W_l such that $i < k < j < l$

13 – Distributing links to multiple planes: an algorithm

Input: $\langle W_1, W_2, ..., W_n \rangle$, and a set of dependency links
Output: a restricted multiplanar representation for the dependency structure

Initially, assert that all the links belong to the plane number 0.
for foreach i in $[1..n-1]$ and j in $[i+1..n]$
  if there is a link connecting words W_i and W_j
   denote the plane where the link between W_i and W_j belongs to with p
   if, there are two linked words W_k and W_l such that $i < k < j < l$
    minimize k and maximize l with respect to k)
    lift each link encompassed in the range $[k..l]$ into the plane $p+1$.

14 – Modeling dependency structures: the evaluation

- Test suite: Danish Dependency Treebank (DDT), 100 000 tokens
- We processed DDT with the presented treebank transformation algorithm
- We examined the resulting representation in order to see how many planes were actually needed and what kind of constructions were involved when more planes were needed.
- Moreover, we developed a very promising local heuristics for proving acyclicity of dependency trees (see the paper)

15 – The results

- Extraposition and certain grammatical constructions often required an additional plane. Some examples:
  - coordination with two particles (like *both ... and*, *nether ... nor*, etc.)
  - constructions like *it was the same year when ...*
- Three planes were enough to model 99.97 % of the corpus (two more complex sentences were found within 5540 sentences)

16 – The linguistic relevance

- The number of required planes correlated with our subjective judgment of the difficulty of the sentence
- Restricted multiplanarity with a fixed number of planes rules out a number of erroneous analyses
- The model captures real examples of cross-serial dependencies in Swiss German.
- A conjecture (not yet formally proven): the model fails to capture many awkward, non-natural languages such as MIX and $a^n b^n c^n$.

17 – Concluding remarks and possible applications

- We provided a linguistically interesting generalization of planarity
- We provided a specification for a grammarless model for dependency structures
- We carried out some tests using the Danish Dependency Treebank

Possible applications:

- treebank validation, data mining, finding non-prototypical analyses,
- a basis for a complexity hierarchy of dependency grammars,
- serves a a model-theoretic account of dependency syntax, and
- dependency syntactic parsing (even with finite-state methods).